# Evaluating Multi-Channel Multi-Device Speech Separation Algorithms in the Wild: A Hardware-Software Solution

Ceolini, Enea ; Kiselev, Ilya ; Liu, Shih-Chii

Abstract: Evaluation methods for multi-channel speech separation algorithms in the real world are becoming increasingly important as the number of applications involving audio assistants and hearing aid devices continues to grow. To make such evaluations easier, this paper presents a multi-microphone hardware platform, WHISPER, built specifically for this purpose and its subsequent use for evaluating speech processing algorithms. The platform can also be constructed as an ad-hoc wireless acoustic sensor network (WASN) with high synchronization precision. Using WHISPER, we describe real-world experiments where an example speech separation algorithm is applied to mixtures of varying number of talkers and signal-to-noise ratios. The results when compared with those from a simulated environment, show the usefulness of WASNs and that simulations tend to underestimate the difficulty of speech separation in real-world scenarios. This work represents an important step towards developing a hardware-software framework for evaluating speech processing algorithms in the wild.

DOI: https://doi.org/10.1109/taslp.2020.2989545

# Evaluating multi-channel multi-device speech separation algorithms in the wild: a hardware-software solution

Enea Ceolini, *Student Member, IEEE,* Ilya Kiselev, *Student Member, IEEE,* and Shih-Chii Liu, *Senior Member, IEEE*

***Abstract*—Evaluation methods for multi-channel speech separation algorithms in the real world are becoming increasingly important as the number of applications involving audio assistants and hearing aid devices continues to grow. To make such evaluations easier, this paper presents a multi-microphone hardware platform, WHISPER, built specifically for this purpose and its subsequent use for evaluating speech processing algorithms. The platform can also be constructed as an ad-hoc wireless acoustic sensor network (WASN) with high synchronization precision. Using WHISPER, we describe real-world experiments where an example speech separation algorithm is applied to mixtures of varying number of talkers and signal-to-noise ratios. The results when compared with those from a simulated environment, show the usefulness of WASNs and that simulations tend to underestimate the difficulty of speech separation in real-world scenarios. This work represents an important step towards developing a hardware-software framework for evaluating speech processing algorithms in the wild.**

***Index Terms*—array signal processing, wireless acoustic sensor networks, speech separation, submicrosecond synchronization.**

## I. Introduction

THE problem of speech separation has widely been studied for applications involving speech enhancement and speech recognition. Different approaches have been proposed over the years with promising results coming most recently from the use of deep learning methods [1]–[3]. Because single-channel demixing by itself is challenging, a better approach to speech separation in a difficult auditory scene (e.g. speech in noise, multiple simultaneous speakers) is the use of multiple microphones in a framework known as multi-channel audio source separation (MASS) [4]–[7]. Indeed, in the last decade, a lot of attention has been placed on multi-channel approaches which are more robust than single-channel solutions [8].

Often, algorithms for speech separation such as beamforming or neural network models are tested either with artificial speech mixtures created from simulated environments [9]–[12] or with speech convolved with room impulse responses (RIRs) recorded in controlled environments [7], [13]–[15]. Typically they are not tested with real-world recordings. Because of this, it is difficult to know how well these algorithms perform in the real world. Moreover, it is hard to assess how the performance of these algorithms would change when implemented on an embedded portable platform. Naturally, not all applications need to address the issue of portability or restricted computational power. Nevertheless, modern applications such as audio assistive devices and hearing aids need to function within these constraints.

The additional step of validating the algorithms in the real world is useful for one to know if a subset of algorithms work better in real world conditions even though they show similar performance in simulations using artificial mixtures. With the increasing applicability of speech separation techniques, knowing which algorithms work best in the wild is important [16]. For manageable testing of these algorithms using a flexible hardware setup, wireless acoustic sensor networks (WASNs) incorporating spatially distributed microphones are especially relevant. These networks offer an advantage in that microphones can be arbitrarily placed to cover a space, however, a major challenge is the online synchronization of audio samples collected from the distributed microphones.

Even though certain speech separation algorithms do not need perfect synchronization between the sampling times of distributed microphones (e.g. beamforming algorithms [17]–[19] can deal with any systematic delay within the analysis window), sampling rate offsets due to internal clock drift on different network modules (therefore their microphones) will lead to degradation of the algorithm output [20]. Some work has been carried out in the field of blind synchronization of asynchronous recordings [21]–[25]. Nevertheless, these approaches usually use offline methods and do not provide a solution for low-latency applications targeted by the framework presented in this work.

This paper introduces work on two fronts. First, we present a speech separation WASN hardware platform called WHISPER that allows researchers to validate MASS algorithms in the wild. Second, we compare the performance of an example speech separation algorithm that could be implemented on this hardware platform against its performance in a similar simulation environment. This platform also allows one to study how the implemented algorithms are impacted by factors such as the level of clock synchronization in the WASN and the maximum computing power available at each network module.

The remainder of the paper is organized as follows: Section II describes the hardware and software components of the platform. Section III presents the methods used to analyze the usability of WHISPER in the context of evaluating an example speaker separation and speech enhancement algorithm in a MASS setting. Section IV presents the results of this algorithm

The authors are with the Institute of Neuroinformatics at the University of Zurich and ETH Zurich, Zurich, CH (email: enea.ceolini,kiselev,shih@ini.uzh.ch)

when applied on data collected in a real-world scenario and compares them against the results obtained from applying this algorithms in a similar simulated setting.

## II. WHISPER FOR A WASN

An important aspect of a wireless sensor network (WSN) is the timing synchronization across modules in the network. Synchronization protocols for distributed sensor networks vary depending on the required synchronization precision needed of the application. The IEEE 1588 standard [26] is an early established standard for synchronizing distributed devices on a local network. Even now, there are hardly any low-cost commercial devices that implement this standard for a WSN.

In wireless networks, developed synchronization protocols deal with sources of error such as the transmit and receive times of the signal, the propagation time and the clock skew and drift across the network modules. Many solutions rely on offline algorithms [20], [22], [23] and, more importantly, they consider model-based approaches to the problem of drifting clocks across modules. Because the estimation is done offline, these algorithms do not allow for real-time implementation of MASS algorithms [27]. Online estimation methods for synchronizing the receivers in a WSN have been suggested, for example, in [28]. This method has some similarities to our synchronization method proposed here but requires estimation of the receivers' positions. In addition, the reported synchronization results are based on numerical experiments and not from a hardware implementation.

In a WASN, because of the lower signal frequencies and the shorter distances between microphones, we can use a simpler synchronization algorithm implemented through commercially available components. We developed an easy-to-build modular multi-channel hardware platform (WHISPER) which allows testing of audio processing algorithms in real time and in the wild. This platform was originally developed as a speech separation front-end for a cognitive-controlled hearing aid system [29], [30], and is intended for use in a medium-sized room (5-10 m on a side). The platform consists of multiple modules that can be placed at arbitrary locations within an area of 3 m x 3 m. Each module can interface up to four microphones (see Figure 1) and has computing capabilities for supporting audio signal processing algorithms similar to [20], [31]–[33]. This platform can be easily scaled, i.e., it can be extended to an arbitrary number of modules. Because of the modular design, the construction of an ad-hoc array of microphones using this platform was made easy.

A key feature of our platform is our developed online complete hardware-based synchronization algorithm that corrects for the frequency and phase offsets between the sampling clocks on the different modules. In this work, we improved the WHISPER platform design of [30] by supporting better quality I2S digital microphones and improving the synchronization algorithm so as to reduce further the phase jitter of the sampling clocks in each module.

### A. The hardware

WHISPER is built using commercial off-the-shelf components for a rapid implementation that avoids the need for

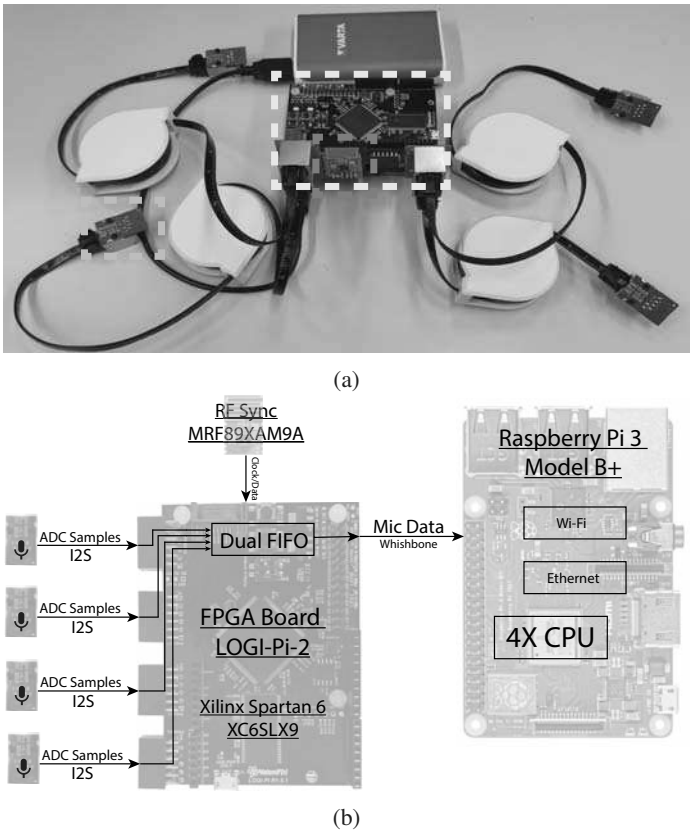



Fig. 1: A WHISPER module. (a) Photograph of WHISPER (yellow box) with four microphones (cyan box) and the RF synchronizer (red box). (b) Hardware component details.

custom hardware development. The two basic building blocks are a CPU-based computing unit, the Raspberry Pi Model 3b+ [1], and a field programmable gate array (FPGA) board, namely the LOGI-PI-2 board from ValentF(x) [2]. The two boards are connected together through a high speed SPI interface. Both computing platforms are necessary, because they provide a different and complementary set of interfaces. The FPGA is used to implement the synchronization algorithm. It is also used to sample the microphones because we can then directly control the sampling clock which is necessary for synchronization purposes. The versatility of the FPGA allows for an easy interface with other digital sensors such as accelerometers [34].

The Raspberry Pi, on the other hand, offers a way of embedding floating point computation, which would be harder to implement on the FPGA. It also provides a set of network connections including Wi-Fi, Bluetooth and Ethernet. The remaining hardware components of WHISPER are the digital microphones and the radio frequency (RF) modules. The digital microphones are 24 bit MEMS microphones from InvenSense (ICS-43432). They have better power supply rejection ratio and higher output resolution compared to the microphones used in the original design [30]. We use only 16 bits of the output to reduce the bandwidth requirements of the

[1] https://www.raspberrypi.org/
[2] http://valentfx.com/logi-pi

platform. The RF transceiver from Microchip Technology Inc. (MRF89XAM9A [3]) is used for implementing the synchronization algorithm described in Section II-C. This module, when operated as a transmitter, can transmit a continuous stream of data. When operated as a receiver, it allows access to the clock recovered from the transmitted data. This recovered clock is necessary to the implementation of our synchronization algorithm. Most of the other data communication modules only provide packetized data therefore timing information is impossible to recover.

### B. The software

The Raspberry Pi on each WHISPER module runs a light version of a Linux OS and has a CPU that supports floating point operations needed for speech processing algorithms [35]. Because algorithms are usually developed on CPUs, the transition step between lab prototyping and deployment of an algorithm in the real world is made easier. The software on WHISPER provides an easy interface to collect the data transmitted via UDP (either via Wi-Fi or via Ethernet Cable) from all modules. This data can include the outputs of MASS algorithms deployed on this platform.

### C. The synchronization algorithm

The challenge to putting a WASN in operation in real-time so that MASS algorithms can be deployed is not trivial. Real-time online synchronization of the clocks on the local modules especially for ad-hoc arrays in unconstrained spaces, is still an active topic of investigation. A short review of various synchronization methods is given in [20]. Highly reliable synchronization solutions that operate across modules in an unconstrained physical space have been proposed in [20], [33] but many of these algorithms rely on offline estimations.

*1) Algorithm description:* Many reported methods use a 2-way message passing algorithm because they estimate the propagation delays between modules in the network in order to synchronize the clocks. Our synchronization algorithm uses only unidirectional message passing where the reference clock is generated by a master clock generator (MCG). This clock is broadcast wirelessly to all the data acquisition modules. The receiving modules adjust the frequency and the phase of their sampling clock to match the received reference clock.

In our target scenario of a medium-sized room (5-10 m on a side), we assume that the modules will be placed within an area of 3 m x 3 m, thus the difference in the radio wave propagation delay is only about 10 ns which is negligible compared to the audio sampling clock period of $\approx 40$ $\mu$s (24 kHz).

The outline of the reference clock recovery algorithm on the receiving module is described next: The first step is to recover the clock from the two signals (RFclock, RFdata) transmitted by the RF module.

In order to estimate the phase and frequency of the reference clock, the RFclock is oversampled by the 120MHz clock on the FPGA (Received clock in Figure 2). The recovered clock has the correct frequency and phase on average but the phase jitter of this clock is too high for our needs (see Figure 3 panel (a).

[3]https://www.microchip.com/wwwproducts/en/MRF89XAM9A

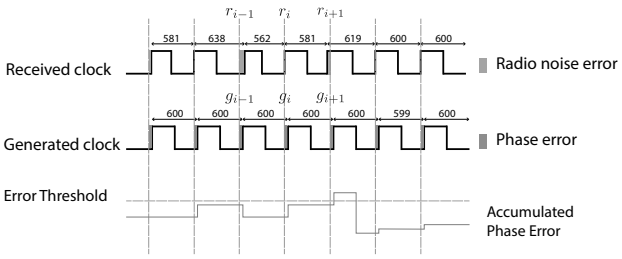



Fig. 2: Signals from the synchronization algorithm implemented on WHISPER. The gray vertical lines delineate the period of the reference clock. The numbers above the waveforms depict the individual cycle lengths measured by the number of FPGA clock cycles.

In an earlier version of our synchronization algorithm [30], the RFclock is used to generate the audio sampling clock directly. The new algorithm described here introduces an additional step which first generates a new clock (Generated clock in Figure 2) that is phase-locked with RFclock but has much less jitter. This clock is then used to generate the audio sampling clock. Because of the reduced jitter, the noise in the sampled audio is decreased with respect to [30].

The synchronization algorithm needs to guarantee that there is no difference in the frequency and little phase difference between the sampling clocks on different modules. Having only a small phase difference between sampling clocks is essential for algorithms that exploit knowledge of the relative positions of audio sources and microphones. These algorithms require implicit spatial location estimation with precision in the order of one degree. In order to fulfill this requirement, we aim to achieve with our synchronization algorithm, an inter-module clock synchronization precision of less than 1 $\mu$s for a sampling frequency of 24 kHz. The synchronization algorithm addresses two problems: The first is to maintain a phase shift of less than one sampling clock cycle between clocks on different modules. The second is to have low clock jitter across time on each module.

To show how synchronization is achieved, we first define $r_i$ and $g_i$ to be the absolute times of the rising edges of the respective received and generated clocks on the $i$th reference clock cycle (see Figure 2 for an example) and further assume that the frequency difference of these clocks is small.

By applying a quadratic loss function to the phase error of the generated clock on the $i$th clock cycle, i.e, $E_i = (r_i - g_i)^2$, the accumulated phase error over the most recent $N$ clock cycles is described by

$$\mathcal{L}_i = \sum_{j=1}^{N} E_{i-j} = \sum_{j=1}^{N} (r_{i-j} - g_{i-j})^2. \quad (1)$$

We then estimate the phase shift, $\epsilon$, needed for the generated clock so that the accumulated phase error $\mathcal{L}$ is minimized:

$$\mathcal{L}_i(\epsilon) = \sum_{j=1}^{N} (r_{i-j} - (g_{i-j} + \epsilon))^2 \quad (2)$$

$$\hat{\epsilon}_i = \underset{\epsilon}{\operatorname{argmin}}(\mathcal{L}_i(\epsilon)) \quad (3)$$

A simple derivation shows that

$$\hat{\epsilon}_i = \sum_{j=1}^{N} (r_{i-j} - g_{i-j})/N \quad (4)$$

This phase shift is simply a moving average of the phase difference between received and generated clocks, which corresponds to a finite impulse response (FIR) filter with the memory length $N$. In order to reduce the memory usage of the filter on an FPGA, we used an infinite impulse response (IIR) filter that has similar characteristics but requires just one register. Simulations show that qualitatively there is no difference in the synchronization outcome. Thus, instead of computing moving average, we compute an exponential moving average of the phase difference over time:

$$\tilde{\epsilon_i} = (r_i - g_i)k_1 + \epsilon_{i-1}(1 - k_1) \quad (5)$$

where $k_1$ is the integration constant chosen to match a decay time equal to $N$ clock cycles. In our experiments, $k_1 = 1/N = 1/2048$.

If this estimated phase shift $\tilde{\epsilon}$ exceeds a half of the FPGA clock cycle, we make a phase correction of the generated clock by one FPGA clock cycle. The correction is done by either decreasing or increasing the length of the next generated clock cycle according to the sign of $\tilde{\epsilon_i}$ over time, therefore the next rising edge of the generated clock is:

$$g_{i+1} = \begin{cases} g_i + \tilde{r}_i + 1, & \tilde{\epsilon}_i > 1/2 \\ g_i + \tilde{r}_i - 1, & \tilde{\epsilon}_i < -1/2 \\ g_i + \tilde{r}_i, & otherwise \end{cases} \quad (6)$$

where $\tilde{r}_i = (r_i - r_{i-1})k_2 + \tilde{r}_{i-1}(1-k_2)$ is an exponential moving average of the received clock period. After the correction is done, the estimation of $\epsilon$ starts anew. We set $k_2 = 1/65536$, that corresponds to a 0.33 s averaging window. The window determines the time taken to track the frequency drift of the RFclock.

This synchronization process is shown in Figure 2. The phase error between the received master clock and the generated clock is accumulated over time. When the accumulated phase error exceeds the threshold, the period of the generated clock is corrected.

The described method provides the sampling clock frequency and phase synchronization. In addition, the absolute time counters at the modules also need to be synchronized in order to properly align in time, the data collected at different modules. For this purpose, the data stream (RFdata) transmitted from the synchronization master is used. The data is transmitted with the bitrate of the reference clock, i.e. 200 kHz. The continuous data stream is logically divided into 1600-bit packets. Each packet has a header and a data payload. The header is comprised of a 13-bit pattern known as a Barker code of length of 13, followed by a 7-bit timestamp. Because of its low-autocorrelation property, the Barker code is used to detect the start of the packet reliably even in poor radio conditions. Each slave module keeps track of the absolute time by counting received reference clock cycles, and also by comparing its own time counter with the timestamp sent by the synchronization master in the packet header. The data payload consists of a sequence of alternating 1's and 0's. The execution of the algorithm on the platform is described by Algorithm 1.

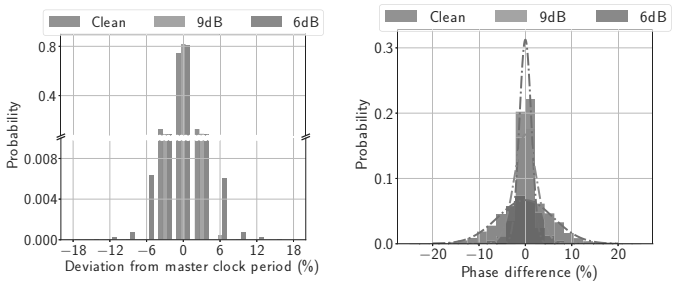
**Algorithm 1** WHISPER synchronization

1: Synchronization master starts continuously transmitting the synchronization data sequence
2: Slave module recovers the master reference clock from the data stream and continuously estimates its period $\tilde{r}_i$
3: Slave module generates its own clock $g_i$ with the period $\tilde{r}_i$ and measures its average phase shift $\tilde{\epsilon}_i$ with respect to the received reference clock $r_i$
4: Slave module compares the average phase shift $\tilde{\epsilon}_i$ with the threshold $\pm 1/2$ and updates the length of the next generated clock cycle $g_{i+1}$ according to Eq. 6
5: The average phase shift $\tilde{\epsilon}_i$ is set to 0 and the step 2 repeats

*2) Measurements:* We show measurement results from experiments that evaluated two aspects of the synchronization algorithm. The first experiment measures the amount of synchronization between the generated clock of a single module and the received master clock. The second experiment measures the synchronization error between the generated clocks on 4 different modules. The error is measured as the phase shift between the clock of a reference module and the clocks of the other 3 modules.

The results from the first experiment are presented in Figure 3. We first show the empirical distribution of the deviation of the received clock period length from the master clock period length over 30E6 clock cycles in subfigure (a). Measurements are done for three conditions, namely in the absence of radio noise (clean), and in the presence of radio noise with two SNR levels (9 dB and 6 dB). The radio noise is generated by another RF transmitter for which we can set the transmission power and thus we can control the SNR. From these curves, we first see that the period deviation is quantized because the clock frequency of the RF module (6.4 MHz) is resampled using the faster FPGA clock which runs at 120 MHz. We also see that the means of the distributions are all around 0 ($\mu^a_{clean} = 0.0001\%$, $\mu^a_{9dB} = 0.0002\%$, $\mu^a_{6dB} = 0.007\%$), while the standard deviations increase from ($\sigma^a_{clean} = 1.35\%$) in the clean case to ($\sigma^a_{9dB} = 1.37\%$) and ($\sigma^a_{6dB} = 1.40\%$) in the 9 dB and 6 dB SNR cases, respectively. The small probabilities associated with the tails of the distributions in the RF noise cases show the effect of the noise but do not compromise the synchronization.

Figure 3 (b) shows the distribution of the phase difference between the generated clock of the slave module and its received reference clock. The shape of this distribution is due primarily to the phase jitter of the received reference clock




(a) (b)

Fig. 3: Deviation of period and phase of the received reference clock in three conditions: clean, RF noise with 9dB SNR and RF noise with 6dB SNR. (a) Deviation of the received clock period with respect to the master clock period. The y-axis is discontinuous in order to show the extremes of the distribution. (b) Phase difference distribution between received and generated clock signals.

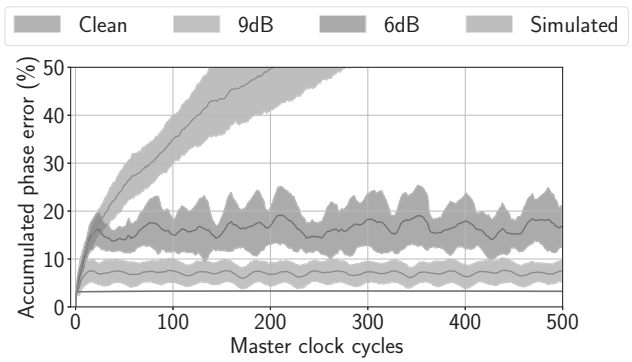



Fig. 4: Accumulated phase error between generated clock and received clock for the simulated case of independent phase error (gray) and measured from the hardware platform in the cases where RF noise is absent (blue) or present with SNR of 9 dB (green) or 6 dB (red).

because the generated clock has substantially lower phase jitter (see Figure 5). Similarly to panel (a), the means of the distributions in panel (b) are around 0 ($\mu^b_{clean} = 0.058\%$, $\mu^b_{9dB} = 0.057\%$, $\mu^b_{6dB} = 0.070\%$), and the standard deviations of the probability distributions increase from ($\sigma^b_{clean} = 1.27\%$) for the clean case to ($\sigma^b_{9dB} = 2.38\%$) and ($\sigma^b_{6dB} = 5.96\%$) for the 9 dB and 6 dB SNR cases, respectively. The results show that, even in noisy conditions when the jitter increases, the synchronization algorithm is successful in keeping the average phase difference between the reference and generated clocks around 0.

From the same experiment, we measured the accumulated phase error between the received reference clock and the generated clock at one module (Figure 4). We can clearly see that in both clean (blue) and noisy condition (green for 9 dB and red for 6 dB), this accumulated phase error saturates quickly. The accumulated phase error curve for the clean case looks flat because the variance is significantly smaller ($\sigma_{clean} = 0.1\%$) than the variance for all the other cases ($\sigma_{9dB} = 2.3\%$, $\sigma_{6dB} = 4.5\%$, $\sigma_{gauss} = 8.7\%$). We compare these measurements with the output of a simulated scenario where the error distribution is the same as in the noisy condition with 9 dB SNR, i.e, a Gaussian distribution with $\mu = 0\%$ and $\sigma = 2.38\%$. In the simulation, the phase errors are independently drawn in sequence. In this case (gray curve in Figure 4), the accumulated phase error grows over time. This difference seen between the curves suggests that the errors in the received clock period of the hardware platform are not independent because probably the errors cancel out over time. This shows that even in noisy conditions the error does not grow indefinitely and can be corrected appropriately.

In the second experiment, we measure the level of synchronization between the generated clocks of four modules placed in different spatial arrangements from recordings of 1 minute. Figure 5 shows the measured phase shift and jitter of the clocks of 3 modules with respect to the clock of a reference module for 4 different spatial arrangements. The recordings are done in 2 rooms. The measured distributions in Figures 5 (a)-(c) come from a room with fewer RF-reflective surfaces than the room used for the measured data shown in Figure 5 (d).

For all spatial arrangements that we tested, the phase shift was less than 200 ns which is well within our specifications for precise localization and beamforming.

The corresponding jitter, as measured by the standard deviation of the phase difference distribution, was less than 9 ns for the spatial arrangements used in Figures 5 (a)-(c). The increased jitter in Figure 5 (d) is very likely due to the increased RF-reflective material in the second room instead of the spatial arrangement of the modules.

The jitter value of around 9 ns (which is $0.025\%$ of the 24kHz sampling clock period) is low enough to guarantee that the sampling noise introduced by the jitter still yields high SNR of the sampled signal [36].

*3) Functionality range:* The results from the experiments in this section indicate that WHISPER is expected to work with most configuration of the nodes within a space of 3 m x 3 m, and in the presence of radio noise down to a SNR of 6 dB. Below these values, the synchronization is not expected to work well, in particular we assessed that synchronization fails in the presence of 3 dB radio noise. The amount of RF reflective material such as large metal and concrete surfaces [37] will also affect the synchronization quality. However, a synchronization signal on the WHISPER module indicates in a short time if the system will work in a chosen space.

## III. Methods

This section describes the methods for testing the speaker separation and speech enhancement algorithms deployed on the platform. It also describes an experimental setup where the performance of a MASS algorithm is validated in a real-world environment.

### A. Real-world setup

Using the WHISPER platform with four modules corresponding to 16 randomly placed microphones, we record an

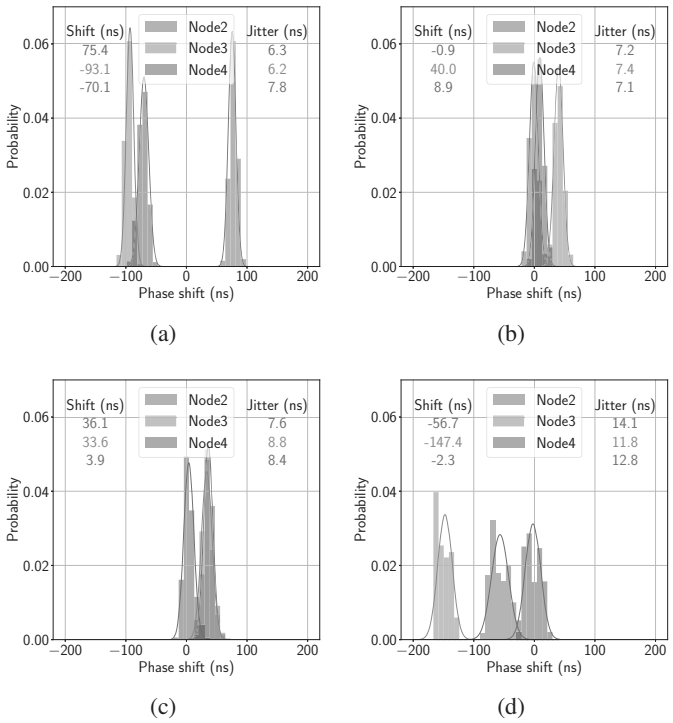



Fig. 5: Distribution of phase differences between the clock of a reference module and the clock of other 3 modules for 4 spatial configurations. (a) Square arrangement with an edge length of 2.5 m. (b) Line arrangement with a spacing of 0.8 m between modules. (c) Random arrangement where all modules are equidistant (1 m) from the transmitter. (d) Random arrangement within a radius of 2 m. The data for panels (a), (b) and (c) was recorded in a room with fewer RF-reflective surfaces than the room in which the data for panel (d) was recorded. The mean ('Shift') and standard deviation ('Jitter') of the distributions are reported within the panels.

extensive dataset of multiple talkers speaking concurrently and also of a single talker speaking in the presence of babble noise at different signal-to-noise ratio (SNR) levels [4]. The recording is done in two separate sessions corresponding to different configurations of microphones and loudspeakers. The minimum and maximum microphone spacing is 10 cm and 2.3 m respectively. The average is $\approx$ 0.5 m. The 5 m x 6 m room used for the recordings has a reverberation time $T_{60} = 0.2s$.

For testing the speaker separation algorithms, we record scenarios with 2, 3 and 4 competing talkers. The talkers' voices are normalized to unit power so that each talker has the same power. For a mixture of 2 talkers, the resulting SNR for a single talker is 0 dB, while for a mixture of 3 talkers, the SNR is $-6$ dB and for a mixture of 4 talkers, the SNR is $-9.5$ dB. We also validated that the average SNR values computed from the microphone recordings are similar to the theoretical SNRs. The talkers are readers of Audiobooks freely available from LibriVox [5].

[4]The dataset is publicly available on Zenodo at https://doi.org/10.5281/zenodo.3688540

[5]https://librivox.org/

Each recording has two phases: A first phase where every talker in the mixture speaks alone for 15 seconds; and a second phase where all talkers speak concurrently for another 15 seconds. The recording of the first phase can be used for the calibration phase of the speaker separation algorithms. The dataset includes 30 recordings for each of the scenarios corresponding to 2, 3 or 4 talkers.

For testing the speech enhancement algorithms, we record a single talker in the presence of diffuse babble noise at SNR levels of $[0, -5, -10]$ dB. As in the speaker separation case, these SNR values are valid on average at all the microphones. Additionally, each recording has two phases: A first phase where the talker speaks in the absence of noise for 15 seconds, and a second phase where the talker speaks in the presence of the noise for another 15 seconds. The dataset includes 30 recordings for each of the three SNR scenarios.

Even though the number of recordings is limited (i.e. 180 recordings for both the separation and enhancement scenarios), the use of 16 microphones allows us to produce more samples with different ad-hoc array configurations. For example, one sample corresponds to 254 recordings if one combines the recordings across all possible subsets of 8 microphone arrangements. This large number of recordings allows us to obtain meaningful statistics on the planned experiments.

### B. Simulated data

A simulation of the real world setup described in Section III-A is also carried out so we can compare the results of the simulations against those of the real-world recordings. In the simulations, the room size is kept the same and the reverberation time is set to the average value measured in the real room using a sound meter sensor [6]. The RIRs for all the combinations of speakers and microphones are simulated with a GPU version of the image method [38]. The simulated speaker separation and speech enhancement scenes are generated for the same speakers and noise conditions of the real-world data. In total, there are 90 samples for the speaker separation task, with 2, 3, and 4 speakers; and 90 samples for the speech enhancement task with SNR values of 0, $-5$ and $-10$ dB.

### C. The challenge of evaluating a speech enhancement algorithm in the wild

Despite the numerous techniques developed for assessing the output speech quality of a speech separation algorithm, it is still challenging to find a universal accepted metric for evaluating the quality of the separate speech signals. One big challenge is that the *quality* of speech is domain and application specific. There are two main methods of assessing speech enhancement success: via speech recognition tasks and via subjective or objective measures. Unfortunately, the results reported by the two methods might be different since the *quality* required by a speech recognition algorithm might not correspond to the *quality* in terms of speech intelligibility for

[6]Brüel & Kjaer Type 2250-S

humans. Since we do not address the problem of speech recognition here, we only take into account objective measures that evaluate the quality of the signal or its intelligibility. We use scale-invariant signal-to-distortion ratio (SI-SDR) to determine signal quality instead of the classic signal-to-distortion ratio (SDR), one of the used BSS evaluation measures [39]. Our choice is motivated by recent findings that SI-SDR mitigates some of the limitations of SDR [40]. In particular SDR allows, during the evaluation, for modifications to the target speech and a frequency weighting which could lead potentially to high scores, but very poor signal quality. Moreover, because of the way it is defined, SDR can be improved by simply rescaling the estimated signal, while SI-SDR guarantees the scale independence of the result. For the speech intelligibility evaluation, we use short-time objective intelligibility (STOI) metrics [41].

Besides the choice of speech metrics, another big challenge is how one can evaluate an algorithm in the real world where ground truth is not available. One could play recordings from a recorded dataset through a loudspeaker, and then compare the separated signal of a particular speech enhancement algorithm with the original ground truth signal in the recordings. However, this evaluation could yield poor results in terms of objective measures [42] due to the interference and signal distortions introduced by the loudspeaker system and the microphones. In addition, the algorithm itself might reduce the objective quality even if the algorithm has effectively cancelled the undesired interference.

In order to minimize the influence of these distortions, we follow the procedure described here: Let $x_a$ and $x_b$ be the two component signals of a mixture for a speech separation scenario. Assuming that the distortions introduced by the loudspeakers and the microphones are captured in $\hat{x}_i$:

$$\hat{x}_i = x_i + \eta_i \quad for \quad i \in \{a, b\} \tag{7}$$

the recorded mixture, $y$, can be described as

$$y = x_a + x_b + \eta_a + \eta_b = \hat{x}_a + \hat{x}_b. \tag{8}$$

When using an algorithm ($F_a$) to estimate $x_a$, $F_a(y) = \tilde{x}_a$, this estimate ($\tilde{x}_a$) will suffer from some error, that is

$$\tilde{x}_a = \hat{x}_a + \mu_a \tag{9}$$

where $\mu_a$ is due to reverberation, leakage from other audio sources or ambient noise. When using $x_a$ as ground truth during evaluation, the full error $\mu_a + \eta_a$ will be attributed to the algorithm which nevertheless can only correct for $\mu_a$ but has no way of correcting for the error $\eta_a$. To avoid the incorrect attribution of the error $\eta_a$ to the algorithm, we first do separate recordings of $\hat{x}_a$ and $\hat{x}_b$. We then record $y$ and after applying our speech separation algorithm, we can directly compare $\tilde{x}_a$ with $\hat{x}_a$ excluding errors that the algorithms cannot correct. While this way of evaluating an algorithm compensates for some errors that the algorithm cannot correct, it is limited by the assumption that $\eta_i$ stays the same across recordings which might not be the case in general. Moreover, this technique cannot be applied when evaluating the dereverberation power of a speech separation algorithm.

### D. Minimum variance beamforming

We describe here experiments for the evaluation of a beamforming algorithm on the multi-microphone WHISPER platform. These experiments will help demonstrate the importance of synchronizing the microphone sampling clocks in ad-hoc multi-microphone arrays and the potential benefits for speech separation. We considered two scenarios: speech separation and speech enhancement. In the first scenario, the goal is to extract a single speaker from multiple concurrent speakers. In the second scenario, a single speaker is talking in the presence of diffused babble noise and the goal is to maximally reduce the noise in order to enhance the speech of the speaker. For both scenarios, we used minimum variance distortionless response (MVDR), a beamforming algorithm which is simple to implement but at the same time fast to execute and robust to reverberant environments [43]. Given that the classic formulation of MVDR is in the frequency domain, we will consider signals in the short-time Fourier transform (STFT) domain.

Consider $j$ point sources and $m$ microphones. Then the signals for time point $n$ and frequency bin $f$ collected by the microphones can be written in matrix form as

$$\mathbf{x}(n, f) = \mathbf{A}(n, f)\mathbf{s}(n, f) + \mathbf{u}(n, f) \tag{10}$$

where $\mathbf{A}(n, f) = [\mathbf{a}_1(n, f), ..., \mathbf{a}_j(n, f)]$ is the mixing matrix of direct paths plus reflections from each source to each microphone, $\mathbf{s}(n, f) = [s_1(n, f), ..., s_j(n, f)]^T$ are the source signals and $\mathbf{u}(n, f)$ corresponds to the diffuse noise source. Without loss of generality, we will consider $s_1(n, f)$ as being the source of interest, rendering $\mathbf{a}_1(n, f)$ the look direction of our beamformer. Note that in the scenario of speech separation, $\mathbf{u}(n, f)$ is 0 as no diffuse noise is present in the scene but only concurrent speakers are present. Conversely, in the scenario of speech enhancement $\mathbf{u}(n, f)$ represents the diffuse noise, while the matrix $\mathbf{A}(n, f)$ reduces to simply the vector $\mathbf{a}_1(n, f)$ since only one source is present in the scene. For sake of clarity, in the reminder of the section we will omit the dependence of the quantities on $(n, f)$.

We compute the beamformer weights $\mathbf{w}$ such that the estimated signal $\hat{d} = \mathbf{w}^H\mathbf{x}$ is as close as possible to the desired signal $s_1$, while maintaining a distortionless response. The optimal weights can be calculated depending on the imposed system constraints. In the MVDR case, we minimize the overall power of the signals while guaranteeing a fixed gain in the direction of our desired signal, through the following optimization:

$$\mathbf{w}_{MVDR} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbf{w}^H \Sigma_{\mathbf{x}} \mathbf{w} \quad s.t. \quad \mathbf{a}_1^H \mathbf{w} = q \tag{11}$$

where $\Sigma_{\mathbf{x}}$ is the sample covariance matrix and $q$ is the gain in the desired direction, usually set to 1. This optimization leads to the optimal weights:

$$\mathbf{w}_{MVDR} = \frac{\Sigma_{\mathbf{x}}^{-1}\mathbf{a}_1}{\mathbf{a}_1^H \Sigma_{\mathbf{x}}^{-1} \mathbf{a}_1} q \tag{12}$$

The algorithm can also be easily implemented in the time domain (e.g. following [44]) depending on the desired latency and computational capability constraints on the platform.

Another important aspect of ad-hoc beamforming is the estimation of the steering vector. While the concept of a steering vector is well defined for an array geometry such as a straight line or a circle, it is not the case for an ad-hoc microphone array. The simplest way of estimating the steering vector is by calculating the principal component of the sample covariance matrix $\mathbf{\Sigma_{As}}$ collected during periods of activity of the desired speaker [45]. Since these periods are generally hard to detect when continuous noise is present, if one knows that only one speaker is present it is easier to use a voice activity detection (VAD) algorithm to compute the covariances of the noise ($\mathbf{\Sigma_u}$) and of the speaker plus noise ($\mathbf{\Sigma_x}$). $\mathbf{\Sigma_{As}}$ can then be estimated following

$$\hat{\mathbf{\Sigma}}_{\mathbf{As}} = \mathbf{\Sigma_x} - \mathbf{\Sigma_u}. \tag{13}$$

In the case of multiple speakers, more sophisticated algorithms, e.g. [42], are needed to estimate and combine the brief periods when a single talker is active. In this work, we do not address the problem of finding periods of activity of the desired speaker, thus we do not utilize a VAD nor the methods described in [42]. Instead, we estimate the covariance matrix $\mathbf{\Sigma_{As}}$ directly from the first 15 seconds (first phase) of each recording as reported in Section III-A.

## IV. Results

This section presents the results from applying MVDR on the dataset described in Section III-A and from simulations described in Section III-B.

These results show how we can easily evaluate a speech separation algorithm for both the simulation and the real world because of the framework of WHISPER.

### A. Experimental setup

For all experiments in this section, the WHISPER platform was used in the configuration described in Section III-A, namely with 4 modules and thus a total of 16 microphones. The modules are connected via a Ethernet cable to a local network which features also a laptop. The microphone data from the modules was transferred and stored on the laptop so that the application of the beamforming algorithm was done offline. The reason for doing the beamforming offline is that we averaged the performance over multiple utterances for many possible configurations of the ad-hoc array in order to obtain meaningful statistics. Since the speech separation was done offline, we did not use a VAD algorithm or the method described in [42] for finding the periods when the speakers were active. Instead, as briefly mentioned in Section III-A, the calculation of the covariance matrix needed to determine the steering vector of the beamformer, was done on the segments with a single speaker. These segments were also used as ground truth for evaluation of the speech quality.

### B. Real world results

*1) Speaker separation:* Speaker separation experiments are carried out for multiple scenarios involving different numbers of concurrent speakers. First, we show results averaged over utterances recorded in all scenarios and analyze the effect of adding microphones to the ad-hoc array. Then we consider separately each scenario with a different number of speakers. In the analysis, we show results for an even number of microphones between two and sixteen.

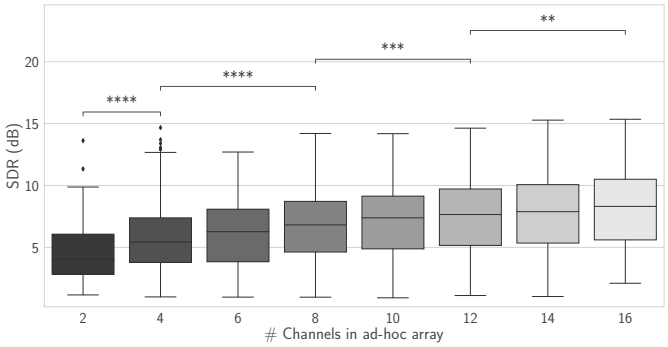



Fig. 6: SDR for speaker separation averaged over all possible numbers of concurrent speakers. Significance has been calculated from a Mann-Whitney statistical test [46] with $p$-values: *($p < 0.05$), **($p < 0.01$), ***($p < 0.001$), ****($p < 0.0001$).

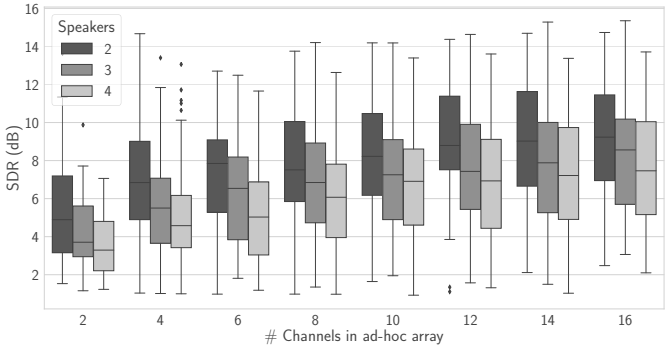



Fig. 7: SDR of separated speech in a speaker separation scenario. Results are for an increasing number of microphones tested on one mixture and across mixtures with 2, 3, and 4 speakers.

We report the statistical significance only for the cases where the number of microphones is a multiple of four, corresponding to an integer number of WHISPER modules. As shown by the SDR results in Figure 6, there is a significant increase in signal quality as more modules (i.e. channels) are included. The effect of increasing the number of microphones while considering different number of speakers in the mixture, is shown in Figure 7. The gain in SDR when adding new modules is more significant for a mixture of four speakers compared to a mixture of two speakers. Theoretically, using more microphones than speakers in a mixture should not give any benefit in the separation. In an ad-hoc arrangement, because of the varying distances of the microphones to the sources, the use of more microphones can statistically lead to an increasing SDR with more channels. However the gains with the use of increasingly more microphones saturate as shown in the figure.

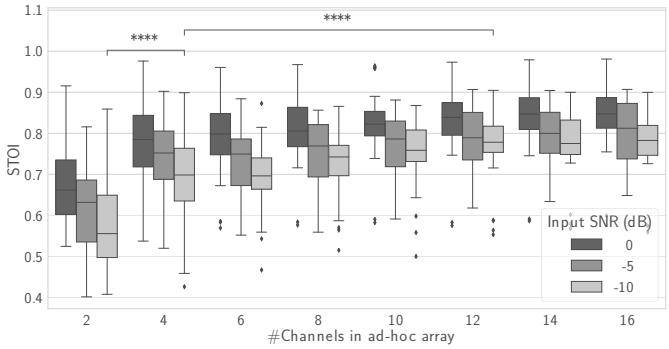



Fig. 8: STOI of enhanced speech in a speech enhancement scenario for an increasing number of microphones and tested across three different input SNRs of speech in noise.

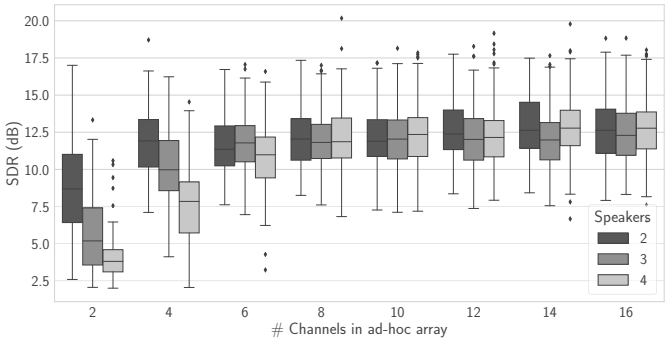



Fig. 9: SDR of the separated speech for speaker separation in a simulated environment. Results are for an increasing number of microphones, and for different speakers in the mixture.

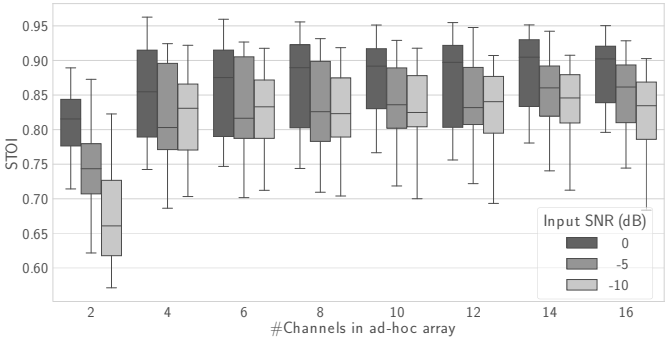



Fig. 10: STOI for the speech enhancement experiment in a simulated environment. Results are for an increasing number of microphones at one SNR, and for different input SNRs.

*2) Speech enhancement:* In these experiments, we extract a single speaker from a mixture and consider the rest as noise. The experiments include scenarios with speech at SNR levels of $[0, -5, -10]$ dB. Similar to the speaker separation experiments, we include the results from using an increasing number of WHISPER modules but instead of using signal quality, we show speech intelligibility which is more important for speech enhancement metrics. Figure 8 shows that the benefits of increasing the number of channels apply not only for the -10 dB case but also for input at 0 dB. Clearly, the gain saturates earlier for the easier scenarios (see the curves for 0 dB SNR), and there is no significant gain with more than 10 microphones. On the contrary, the benefits are more remarkable for the most difficult noisy conditions. Even with a low input SNR (-10 dB), using 3 WHISPER modules (12 microphones) leads to a significant gain of 50% in speech intelligibility metrics compared to the use of only two microphones.

### C. Simulation results

For both the speaker separation and speech enhancement tasks, we simulated the data recording process and applied the MVDR beamforming by randomizing the subsets of ad-hoc arrays in the same way used for the real-world recordings in the previous section. The results are averaged over the whole simulated dataset which contains the same number of samples as the real-world dataset.

*1) Speaker Separation:* Figure 9 shows the SDR for the speaker separation task in the simulated environment for different number of microphones in the ad-hoc array and different numbers of speakers in the mixture. Similar to the real-world results, we can see the saturation effect on the SDR with increasing number of microphones in the array. In particular, we see that the saturation happens earlier, i.e. with fewer microphones, for the simulated scenario compared to the real world scenario. This shows how in theory MVDR needs at least an equal number of microphones to the number of speakers in order to be effective, but in practice, the number of microphones needed to reach a certain SDR is higher. Furthermore, it is worth noting that in the simulated data, the results with a large number of microphones ($> 8$) show that there is no clear difference in SDR between the mixtures which have different number of speakers. In contrast, the results on real-world data show that adding more speakers in the mixture increases the difficulty of the task as seen by the decreasing SDR.

*2) Speech Enhancement:* The results for the simulated task of speech enhancement are depicted in Figure 10 where we considered speech intelligibility as the evaluation metric. Similar to the speaker separation results, the saturation of the STOI gain with increasing number of microphones happens with fewer microphones in the simulations with respect to the real-world environment. This shows that in practical scenarios of speech in noise, in order to obtain a satisfactory speech intelligibility level, one needs more microphones than the number indicated in simulations. Furthermore, simulations results show that with enough microphones ($> 8$) there is no significant difference for scenarios at $0\,\mathrm{dB}$ and -10 $\mathrm{dB}$ SNR. In practice this is not the case, since even for a large number of microphones the scenario with -10 $\mathrm{dB}$ SNR is much harder than the scenario with $0\,\mathrm{dB}$ SNR.

### D. Synchronization in real world

The dataset described in Section III-A was collected during two sessions, each lasting around 4 hours. In 5% of the recordings, some of the modules lost synchronization. We estimated the synchronization error in the recordings by using segments

| Sync modules / total modules | Number mics | Sync error (us) | Task | SDR (dB) | STOI |
|---|---|---|---|---|---|
| 2/2 | 8 | $< 0.1$ | Separation | $7.28 \pm 1.43$ | $0.78 \pm 0.02$ |
| | | | Enhancement | $3.15 \pm 0.65$ | $0.68 \pm 0.05$ |
| 2/3 | 12 | 66 | Separation | $1.44 \pm 2.73$ | $0.482 \pm 0.107$ |
| | | 62 | Enhancement | $0.085 \pm 0.115$ | $0.53 \pm 0.02$ |
| 2/4 | 16 | 30 | Separation | $-1.35 \pm 1.05$ | $0.28 \pm 0.06$ |
| | | 45 | Enhancement | $-1.1 \pm 1.0$ | $0.035 \pm 0.455$ |
| 3/3 | 12 | $< 0.1$ | Separation | $8.75 \pm 1.46$ | $0.767 \pm 0.078$ |
| | | | Enhancement | $3.6 \pm 0.4$ | $0.735 \pm 0.005$ |
| 3/4 | 16 | 66 | Separation | $2.65 \pm 2.26$ | $0.485 \pm 0.112$ |
| | | 82 | Enhancement | $0.185 \pm 0.065$ | $0.595 \pm 0.035$ |
| 4/4 | 16 | $< 0.1$ | Separation | $9.6 \pm 1.85$ | $0.804 \pm 0.0206$ |
| | | | Enhancement | $5.41 \pm 1.03$ | $0.811 \pm 0.004$ |

TABLE I: Impact of synchronization error between modules of different WHISPER platform configurations. Results are reported for SDR and STOI for both speech separation and speech enhancement tasks. The average SNR is -3 dB for the separation task (scenarios with 2 and 3 competing speakers) and -6 dB for the enhancement task (scenarios with SNRs of 0, -5 and -10 dB).

where synchronization failed and those where synchronization was successful. We then computed the cross correlation between the recordings of two microphones and the difference in the peak times between the synchronized and desynchronized cases represents the estimated synchronization error. Using these recordings, we analyzed how failed synchronization and the different number of synchronized modules affect the speech separation quality. We estimated the synchronization error and evaluated the SDR and STOI on single recordings of 15 s and then averaged the results. A summary of these results is given in Table I. The impact of the synchronization error on the SDR and STOI of both speech enhancement and speaker separation can clearly be seen.

The separation quality increases with the use of more microphones as seen from the results for the 2/2, 3/3, and 4/4 cases (first column). This finding is in line with the results in Section IV-B. The other configurations represent cases where one or more modules are desynchronized. For these cases, Table I reports the average synchronization error in the recordings. Note that these are the cases when the synchronization fails. In the case where 2 out of 4 modules are desynchronized, we see the worst results with SDR$<$0, which represents a failure in speaker separation or enhancement. In the case when 3 out of 4 modules are synchronized, the increased SDR of around 3 dB shows that increasing the number of synchronous modules improves the SDR even when some of the modules are desynchronized.

## V. Discussion

We describe the multi-microphone WHISPER platform useful for testing algorithms in the wild and present comprehensive results for a particular speech separation algorithm, namely MVDR, using this platform. We compare the results from running a MVDR beamformer on WHISPER in a real-world environment versus the equivalent in a simulated environment. The results from both environments are in general agreement but with some difference in the speech separation quality as a function of the number of microphones. First, the number of microphones needed to achieve a desired output SDR is higher in the real-world scenario than in the simulated world. Second, there is a slower saturation of the SDR output for the real-world environment for increasing number of microphones compared to the simulated environment. These results show that simulations tend to underestimate the difficulty of speech separation in real-world scenarios and, more importantly, underestimate the usefulness of WASNs of which WHISPER is a perfect example.

The overall 3 dB SDR difference between the simulated and real-world environments is due to a number of factors which the simulation does not take into account. First, the furniture in the room creates reflections, in particular, the table on which the microphones and the speakers are placed. This furniture is not present in the simulated environment. Second, the noise of the microphone electronics introduces distortions in both the ground truth and the mixture. Last but not least, we have perfect ground truth for the simulated case while we do not have it for the real-world experiments. The closest to ground truth is the recordings when the speaker is talking alone. Nevertheless, we do not have a perfect alignment between ground truth and separated speech. Since the estimation of the alignment might not be precise, this further introduces errors and reduces the SDR which is calculated in a pointwise manner.

The WHISPER platform has been used to implement various other speech processing algorithms which have different memory and computational requirements. It was used as a speech separation front-end during a demonstration for a hearing aid application [29]. In this work, a set of well-established algorithms including MVDR, speech-distortion weighted multi-channel Weiner filter (SDW-MWF), maximum SNR (MSNR), mask-based MVDR (MB-MVDR) and mask-based generalized eigenvalue (MB-GEV) was tested on this platform. These beamforming algorithms have very small memory requirements (in the order of tens of MB) and can

easily run in real time on one WHISPER module. The platform was also used to evaluate a more complex speech enhancement algorithm which combined both an MVDR beamformer and a deep neural network (DNN) for estimating the speech and noise mask [47]. This algorithm had higher memory requirements than the simple beamformers due to the additional DNN. The memory requirement was between 30 to 80 MB depending on the network size. In general, this platform can run any speech separation algorithm that fits within the 1 GB of available RAM on the Raspberry Pi of each WHISPER module.

## VI. Conclusion

As practical audio assistant products become more prevalent in the home and workplace, the evaluation of those speaker separation and speech enhancement algorithms that function well in the real world is becoming important. Developments in this area are also useful for hearing aid research. The WHISPER platform developed originally for a cognitive-controlled hearing aid system [29] can also be interfaced to other devices such as a camera [48], accelerometer [34], and EEG amplifiers [49], that can be used to support multi-modal speaker separation and speech enhancement algorithms. It has been used to implement multiple speech separation algorithms that run in real-time [29], [47], [50]. The evaluation results of an example speaker separation and beamforming algorithm presented in this work, demonstrate that the platform can be effectively used to evaluate MASS algorithms in the real world.

## Acknowledgments

We acknowledge Daniel Wong and Jens Hjortkjær for discussions on the use of the platform and Adrian Huber for discussions on the synchronization algorithm. This work was partially supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 644732 and the Swiss National Science Foundation grant agreement No. 200021_172553.

## References

[1] Y. Luo and N. Mesgarani, "TaSNet: Time-domain audio separation network for real-time, single-channel speech separation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 696–700.

[2] J. R. Hershey, Z. Chen, J. L. Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 31–35.

[3] J. Chen and D. Wang, "Long short-term memory for speaker generalization in supervised speech separation," *The Journal of the Acoustical Society of America*, vol. 141, no. 6, p. 4705, 2017.

[4] K. Reindl, S. Meier, H. Barfuss, and W. Kellermann, "Minimum mutual information-based linearly constrained broadband signal extraction," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, pp. 1096–1108, 2014.

[5] H. Barfuss, M. Mueglich, and W. Kellermann, "HRTF-based robust least-squares frequency-invariant polynomial beamforming," in *2016 IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2016, pp. 1–5.

[6] L. Zhao, J. Benesty, and J. Chen, "Design of robust differential microphone arrays," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, pp. 1455–1466, 2014.

[7] O. Schwartz, S. Gannot, and E. A. P. Habets, "Multispeaker LCMV beamformer and postfilter for source separation and noise reduction," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 940–951, 2017.

[8] S. Gannot, E. Vincent, S. Markovich-Golan, and A. Ozerov, "A consolidated perspective on multimicrophone speech enhancement and source separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 692–730, April 2017.

[9] D. Levin, E. A. P. Habets, and S. Gannot, "Robust beamforming using sensors with nonidentical directivity patterns," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 91–95.

[10] O. Thiergart, M. Taseska, and E. A. P. Habets, "An informed parametric spatial filter based on instantaneous direction-of-arrival estimates," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 2182–2196, Dec 2014.

[11] C. A. Anderson, P. D. Teal, and M. A. Poletti, "Spatially robust far-field beamforming using the von Mises(-Fisher) distribution," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2189–2197, Dec 2015.

[12] A. R. Moghimi and R. M. Stern, "An analysis of binaural spectro-temporal masking as nonlinear beamforming," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 835–839.

[13] H. Barfuss, C. Huemmer, G. Lamani, A. Schwarz, and W. Kellermann, "HRTF-based robust least-squares frequency-invariant beamforming," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2015, pp. 1–5.

[14] X. Zhang and D. Wang, "Deep learning based binaural speech separation in reverberant environments," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 5, pp. 1075–1084, May 2017.

[15] Y. Yu, W. Wang, and P. Han, "Localization based stereo speech source separation using probabilistic time-frequency masking and deep neural networks," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2016, no. 1, p. 7, Mar 2016.

[16] L. Girin, S. Gannot, and X. Li, "Chapter 3 - audio source separation into the wild," in *Multimodal Behavior Analysis in the Wild*, ser. Computer Vision and Pattern Recognition, X. Alameda-Pineda, E. Ricci, and N. Sebe, Eds. Academic Press, 2019, pp. 53 – 78.

[17] S. Gannot, D. Burshtein, and E. Weinstein, "Signal enhancement using beamforming and nonstationarity with applications to speech," *IEEE Trans. Signal Processing*, vol. 49, pp. 1614–1626, 2001.

[18] S. Doclo and M. Moonen, "GSVD-based optimal filtering for single and multimicrophone speech enhancement," *IEEE Trans. Signal Processing*, vol. 50, pp. 2230–2244, 2002.

[19] E. Warsitz and R. Häb-Umbach, "Blind acoustic beamforming based on generalized eigenvalue decomposition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 1529–1539, 2007.

[20] J. Schmalenstroeer, P. Jebramcik, and R. Häb-Umbach, "A combined hardware-software approach for acoustic sensor network synchronization," *Signal Process.*, vol. 107, no. C, pp. 171–184, Feb 2015.

[21] M. H. Bahari, A. Bertrand, and M. Moonen, "Blind sampling rate offset estimation for wireless acoustic sensor networks through weighted least-squares coherence drift estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 3, pp. 674–686, March 2017.

[22] D. Cherkassky and S. Gannot, "Blind synchronization in wireless acoustic sensor networks," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 25, no. 3, pp. 651–661, Mar. 2017.

[23] C. Llerena, R. Gil-Pita, D. Ayllón, H. A. Sánchez-Hevia, I. Mohino-Herranz, and M. Rosa, "Synchronization for classical blind source separation algorithms in wireless acoustic sensor networks," in *2016 IEEE Statistical Signal Processing Workshop (SSP)*, June 2016, pp. 1–5.

[24] K. Ochi, S. Miyabe, and S. Makino, "Multi-talker speech recognition based on blind source separation with ad-hoc microphone array using smartphones and cloud storage." in *Interspeech*, 2016, pp. 3369–3373.

[25] S. Araki, N. Ono, K. Kinoshita, and M. Delcroix, "Meeting recognition with asynchronous distributed microphone array using block-wise refinement of mask-based MVDR beamformer," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5694–5698.

[26] I. S. Association *et al.*, "1588-2008—IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *International Standard*, 2008.

[27] J. Schmalenstroeer and R. Häb-Umbach, "Efficient sampling rate offset compensation - an overlap-save based approach," in *2018 26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 499–503.
[28] D. Zachariah, S. Dwivedi, P. Händel, and P. Stoica, "Scalable and passive wireless network clock synchronization in LOS environments," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3536–3546, June 2017.
[29] D. D. E. Wong, J. Hjortkjær, E. Ceolini, S. V. Nielsen, S. R. Griful, S. Fuglsang, M. Chait, T. Lunner, T. Dau, S.-C. Liu, and A. de Cheveigné, "A closed-loop platform for real-time attention control of simultaneous sound streams," in *ARO Midwinter meeting (abstract)*, vol. ARO Midwinter meeting (abstract), 2018.
[30] I. Kiselev, E. Ceolini, D. Wong, A. d. Cheveigne, and S.-C. Liu, "WHISPER: Wirelessly synchronized distributed audio sensor platform," in *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, Oct 2017, pp. 35–43.
[31] L. Girod, M. Lukac, V. Trifa, and D. Estrin, "The design and implementation of a self-calibrating distributed acoustic sensing platform," in *2016 4th International Conference on Embedded Networked Sensor Systems*, 2006, pp. 71–84.
[32] M. Quintana-Suárez, D. Sánchez-Rodríguez, I. Alonso-González, and J. Alonso-Hernández, "A low cost wireless acoustic sensor for ambient assisted living systems," *Applied Sciences*, vol. 7, no. 9, p. 877, 2017.
[33] H. Afifi, J. Schmalenstroeer, J. Ullmann, R. Häb-Umbach, and H. Karl, "Marvelo: A framework for signal processing in wireless acoustic sensor networks," in *Speech Communication; 13th ITG-Symposium*. VDE, 2018, pp. 1–5.
[34] M. Zohourian and R. Martin, "Binaural speaker localization and separation based on a joint ITD/ILD model and head movement tracking," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 430–434.
[35] J. Eldon and C. Robertson, "A floating point format for signal processing," in *1982 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 7, May 1982, pp. 717–720.
[36] J. R. Higgins, *Sampling theory in Fourier and signal analysis: foundations*. Oxford University Press on Demand, 1996.
[37] I. T. U. (ITU), "Effects of building materials and structures on radiowave propagation above about 100 MHz," Tech. Rep., 07 2015.
[38] D. Diaz-Guerra, A. Miguel, and J. R. Beltrán, "gpuRIR: A python library for room impulse response simulation with GPU acceleration," *CoRR*, vol. abs/1810.11359, 2018.
[39] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, July 2006.
[40] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR half-baked or well done?" in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 626–630.
[41] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of timefrequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, Sep. 2011.
[42] E. Ceolini, J. Anumula, A. E. G. Huber, I. Kiselev, and S.-C. Liu, "Speaker activity detection and minimum variance beamforming for source separation," in *Interspeech*, 2018.
[43] S. Doclo, S. Gannot, M. Moonen, and A. Spriet, "Acoustic Beamforming for Hearing Aid Applications," in *Handbook on Array Processing and Sensor Networks*. Hoboken, NJ, USA: John Wiley & Sons, Inc., apr 2010, pp. 269–302.
[44] M. R. Bai, J. Ih, and J. Benesty, "Time-domain MVDR array filter for speech enhancement," in *Acoustic Array Systems: Theory, Implementation, and Application*. Hoboken, New Jersey: John Wiley & Sons, 2013, ch. 7, pp. 287–313.
[45] E. Sarradj, "A fast signal subspace approach for the determination of absolute levels from phased microphone array measurements," *Journal of Sound and Vibration*, vol. 329, no. 9, pp. 1553 – 1569, 2010.
[46] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," in *The Annals of Mathematical Statistics*, 1947.
[47] E. Ceolini and S.-C. Liu, "Combining deep neural networks and beamforming for real-time multi-channel speech enhancement using a wireless acoustic sensor network," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, Oct 2019, pp. 1–6.
[48] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to listen at the cocktail party: a speaker-independent audio-visual model for speech separation," *ACM Trans. Graph.*, vol. 37, pp. 112:1–112:11, 2018.
[49] N. Das, S. V. Eyndhoven, T. Francart, and A. Bertrand, "Adaptive attention-driven speech enhancement for EEG-informed hearing prostheses," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2016, pp. 77–80.
[50] E. Ceolini, J. Anumula, S. Braun, and S.-C. Liu, "Event-driven pipeline for low-latency low-compute keyword spotting and speaker verification system," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 7953–7957.

**Enea Ceolini** (S'17) received his B.Sc. degree in Electrical Engineering and Information Technology at the University of Padova in 2013 and his M.Sc. in Neural System and Computation from the University of Zurich and ETH Zurich in 2016. He is currently a Ph.D. candidate at the University of Zurich and ETH Zurich working on signal processing and machine learning for traditional and event based audio processing.

**Ilya Kiselev** (S'16) received his specialist degree in Physics at the Tambov State University (Russia) in 2000 and his M.Sc. in Applied Mathematics and Physics from the Moscow Institute of Physics and Technology in 2002. He is currently a Ph.D. candidate at the University of Zurich and ETH Zurich working on hardware for signal acquisition and processing for traditional and event-based audio processing.

**Shih-Chii Liu** (M'02 - SM'07) studied electrical engineering as an undergraduate at the Massachusetts Institute of Technology and received her PhD degree in Computation and Neural Systems from the California Institute of Technology, Pasadena, in 1997. She worked at various companies, including Gould American Microsystems, LSI Logic, and Rockwell International Research Labs. Currently, she is a professor with the Institute of Neuroinformatics at the University of Zurich, Switzerland. Her interests include low-power neuromorphic event-driven sensor design; bio-inspired and deep learning algorithms and hardware for energy-efficient, real-time, adaptive intelligent systems.